MILAGRO: A Parallel Implicit Monte Carlo Code for 3-D Radiative Transfer (U)

T.M. Evans and T.J. Urbatsch
X–TM, MS D409, Los Alamos National Laboratory, Los Alamos, NM 87545

We have developed MILAGRO, a general-purpose Implicit Monte Carlo (IMC) code that provides parallel, three-dimensional XYZ, one-group radiative transfer capability. It is written in object-oriented C++ in a maintainable and extensible way. New items in MILAGRO are parallel reproducibility, generic classes templated on Mesh Type (MT), C++/Fortran90 interfacing, and a parallel strategy. Verification of MILAGRO includes coding according to Design by Contract™ (DBC) and successful completion of several test problems. The test problems are variants of the Marshak Wave test problem, Su and Olson's non-equilibrium benchmark [1, 2], and several simple and necessary, but not sufficient, test cases (U).

Keywords: Implicit Monte Carlo, radiation transfer, parallel computing, non-linear transport, object-oriented scientific computing

Introduction

We have developed MILAGRO, a parallel, multidimensional, Implicit Monte Carlo (IMC) code to solve radiative transfer problems. Specifically, as part of the Transport Methods group in X-division (XTM), we endeavored to satisfy three goals with this effort. First, we needed to provide a parallel, 3-D, IMC transport capability for radiation transport problems for use on ASCI computer systems. Second, we needed a new, modern testbed code for transport methods research. Finally, we desired to test new, modern concepts of code-development, parallel scientific programming, and object-oriented and generic code design. From the beginning, MILAGRO was designed with these objectives in mind.

The result is a parallel code written in C++ that is designed to be both maintainable and extensible. Thus, we are assured that we can incorporate new features to satisfy both our customers and research needs required for methods development. At present, MILAGRO supports XYZ and XY orthogonal-structured meshes. The latter is primarily for debugging and testing. MILAGRO runs parallel using both full mesh replication and full mesh domain decomposition. Other plans, described later, are in development. MILAGRO is one-group (grey), but, its design will support multi-group transport in the future.

This paper will describe MILAGRO and the JAYENNE program. We will review the methodologies used to build MILAGRO. We will describe new methods that depart from previous IMC packages. We conclude with our verification suite of test problems that show the correctness of MILAGRO.

JAYENNE Program

The JAYENNE PROGRAM is the umbrella title for XTM's IMC efforts. We have developed a series of codes within JAYENNE leading up to MILAGRO. The first two test

Table 1: Capabilities and features of the MILAGRO and MILSTONE packages.

Package	Host	Geometry	Mesh Type	Parallelism
MILAGRO	stand-alone	XYZ, XY	orthogonal-structured	replication, domain-decomposition
MILSTONE	RAGE	XYZ	orthogonal-AMR	domain-decomposition

codes, MCTEST and IMCTEST, were component tests of our IMC library. Specifically, we used these codes to verify our object-oriented design and to study C++ compilers. The primary result of these efforts was our selection of the KAI KCC compiler [3]. KCC is the only existing compiler that supports most features of the newly adopted ANSI C++ standard [4]. KCC is available on all platforms of interest to ASCI (IBM, SUN, SGI, LINUX, HP, DEC); thus, portability is straightforward because the same compiler is used on different platforms.

At present, JAYENNE contains two major code efforts, MILAGRO and MILSTONE. MILAGRO is our stand-alone IMC code. MILSTONE is an IMC package interfaced with RAGE from the CRESTONE project. As described below, the core IMC components are built in XTM's DRACO system. Thus, MILAGRO and MILSTONE represent the interfaces to different hosts as summarized in Table 1. In other words, both MILAGRO and MILSTONE use the same generic IMC libraries; the fundamental difference is the interface. By using a core set of IMC components that are templated on different Mesh Types (MT), we are able to provide IMC packages quickly to a variety of customers.

Methodologies

Fleck and Cummings Method. The equations we are solving are the *equation of transfer* and the *energy balance equation* [5]. The one-group equation of transfer, neglecting scattering, is

$$\frac{1}{c}\frac{\partial}{\partial t}I + \mathbf{\Omega}\cdot\nabla I = \sigma(\mathbf{x},T)\ [B(T) - I(\mathbf{x},\mathbf{\Omega},t)]\ , \tag{1}$$

where $I \equiv I(\mathbf{x},\mathbf{\Omega},t)$ is the specific intensity, B is the Planck function, σ is the opacity, and T is the material temperature. Equation (1) is correct in the limit of local thermodynamic equilibrium (LTE). The full description of the radiation transfer process requires coupling to the material, whose energy balance is given by

$$\frac{\partial}{\partial t}\mathcal{E}(\mathbf{x},T) = \int_{4\pi}\sigma(\mathbf{x},T)\ [I(\mathbf{x},\mathbf{\Omega}',t) - B(T)]\ d\mathbf{\Omega}'\ , \tag{2}$$

where $\mathcal{E}$ is the material energy density.

We use the IMC algorithm developed by Fleck and Cummings [6]. We will briefly go through the derivation of their algorithm. First, we identify the equilibrium radiation energy density, ϕ,

$$\phi = aT^4\ , \tag{3}$$

where a is the radiation constant. We also note that, in equilibrium, $I = B = \frac{c}{4\pi}\phi$ and, therefore,

$$\int B(T)\,d\mathbf{\Omega} = c\phi(T) = caT^4\ . \tag{4}$$

The nonlinearity in the problem is captured with the variable β, which is defined as follows,

$$\frac{\partial \mathcal{E}}{\partial \phi} = \frac{1}{\beta} \,. \tag{5}$$

Using the preceding definitions in the radiation and material equations, Eqs. (1) and (2), we obtain the following modified equations,

$$\frac{1}{c}\frac{\partial}{\partial t} I + \mathbf{\Omega} \cdot \nabla I = \sigma(\mathbf{x}, T) \, \left[\frac{c}{4\pi} \phi(T) - I(\mathbf{x}, \mathbf{\Omega}, t) \right] \,, \tag{6}$$

$$\frac{\partial \phi}{\partial t} = \beta \int_{4\pi} \sigma(\mathbf{x}, T) \, \left[I(\mathbf{x}, \mathbf{\Omega}', t) - \frac{c}{4\pi} \phi(T) \right] \, d\mathbf{\Omega}' \,. \tag{7}$$

In the modified radiation equation, we want a time-centered expression for the radiation energy,

$$\tilde{\phi} = \alpha \phi^{n+1} + (1 - \alpha) \phi^n \,, \tag{8}$$

where ϕ^n is the radiation energy density at the beginning of a timestep, ϕ^{n+1} is the radiation energy density at the end of a timestep, and α is a user-defined variable specifying the implicitness: $\alpha = 0$ is fully explicit and $\alpha = 1$ is "fully" implicit. (Because the Fleck and Cummings method employs quantities evaluated at the beginning of the timestep, it is not fully implicit [7].) We substitute the time-centered radiation equation for ϕ on the right-hand-side of the modified material equation, Eq. (7), and discretize in time by integrating over a timestep, $t_n \leq t \leq t_{n+1}$. Next, we solve this equation for ϕ^{n+1} and substitute it back into our expression for the time-centered radiation energy density, Eq. (8). This new expression for $\tilde{\phi}$ is used in the modified radiation equation. Lastly, the time-centered radiation intensity is replaced with the instantaneous radiation intensity producing the following radiation equation:

$$\frac{1}{c}\frac{\partial I}{\partial t} + \mathbf{\Omega} \cdot \nabla I + \sigma I = \frac{1}{4\pi}(1 - f)\sigma \int I \, d\mathbf{\Omega}' + \frac{1}{4\pi} f \sigma c \phi^n \,. \tag{9}$$

Note that the total opacity has been divided into an effective absorption, $f\sigma$, and an effective scattering, $(1 - f)\sigma$, using the Fleck factor, f,

$$f = \frac{1}{1 + \alpha \beta c \Delta t \, \sigma} \,. \tag{10}$$

Finally, the material energy density is updated using conservation of energy,

$$\mathcal{E}^{n+1} = \mathcal{E}^n + f \int \int \sigma I \, d\mathbf{\Omega} \, dt - f c \sigma \, \Delta t \, \phi^n \,. \tag{11}$$

We also note that the representation of the Planckian in the Fleck and Cummings method,

$$B \equiv \frac{\tilde{\phi}}{\phi_n} B_n \,, \tag{12}$$

is merely a scaling of the Planckian at the beginning of the timestep [7]. This scaling provides numerical stability, but, in multi-group, does not account for any spectral change.

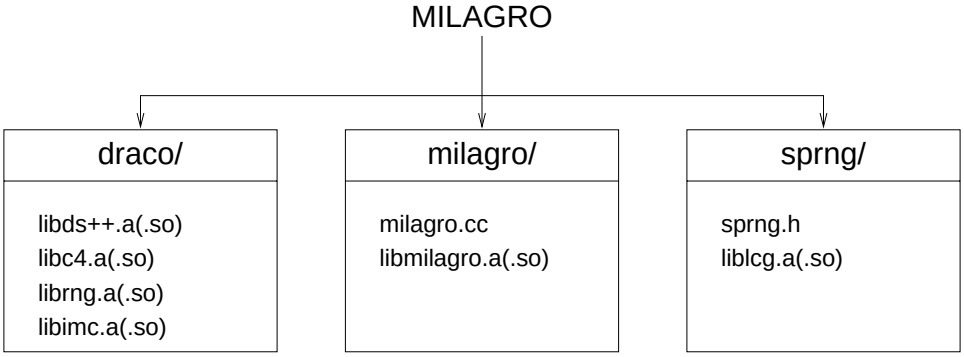

Figure 1: Relationship between DRACO and MILAGRO.

Object-Oriented/Generic Design. MILAGRO is written using an extensible, object-oriented/generic design. This facilitates easy interfacing with multiple hosts. It also allows for a high degree of extensibility. As stated earlier, MILAGRO is built using the DRACO transport system. DRACO is a component-based architecture that provides class libraries geared to numerical physics and transport applications. The guiding concept of DRACO is parameterization on Mesh Types (MT). Thus, packages designed within DRACO simply require MTs, provided by the host, to work in any framework.

Reusable IMC components are built within DRACO. The relationship between DRACO and MILAGRO is illustrated in Fig. 1. DRACO provides a number of service libraries including a communication package (C4), Smart Pointers, numerical containers, and a random number package.

The random number package (RNG) in DRACO provides a set of wrappers used to control the SPRNG [8] random number library. RNG provides a controller that is used to generate random number objects with independent streams. The random number objects are reference counted to allow copying and persistence for message-passing and restarts. These RNG objects facilitate parallel reproducibility as described in the next section.

The IMC package provides five key classes: Source, Particle, Tally, Mat_State, and Opacity. Because these classes are templated on MT, the IMC package can theoretically run on any mesh in any geometry. A large number of service classes are also provided including a Communicator for particle message-passing and a Particle_Buffer container that facilitates particle persistence for communication and restarts. A generalized Transport class controls the interaction of these units as illustrated in Fig. 2.

A final key component in the IMC packages is the use of Design by Contract™[1] (DBC) [9]. Assertions are added to each service in the IMC package to test initial conditions, execution, and final output. Design-by-Contract really is a contract that delineates input to and output from a function. Thus, both parties–the function and the caller–get what they expect. Use of this technology has resulted in higher confidence in the code and reduced debug times.

The DBC assertions are controlled by a compile-time flag. Three levels of DBC support can be switched on and off at compile time. Thus, production versions of the code do not suffer a performance hit from DBC. In addition, an *Insist* function is provided that remains on at all times. This function is useful for checking user input and single execution modules.

[1] "Design by Contract" is a trademark of Interactive Software Engineering.

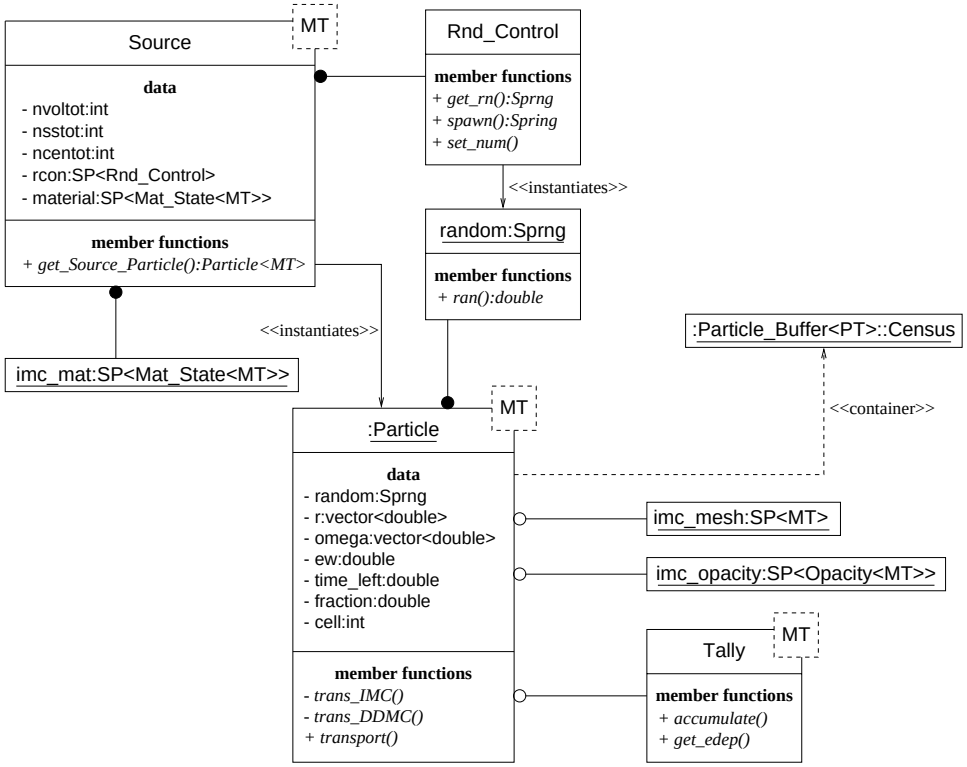

Figure 2: Interaction of primary IMC components.

Parallel Strategy. The natural way to parallelize a Monte Carlo calculation is to repeat the mesh on multiple processors, split the particles up among those processors, and gather their results at the end of calculation. We call this method "full replication" because the mesh is fully replicated on each processor. Full replication is very efficient and gives almost perfect speedup. Often, though, the problems we want to run are so large or so highly resolved that we cannot fit the entire mesh on a single processor in a parallel, shared memory environment. These problems require some sort of domain decomposition in which the mesh is split up among the processors.

The SGI/Cray ASCI machines at Los Alamos are made up of distributed memory boxes, each with several processors (64, 128, or 256) and an effective shared memory (with non-uniform memory access). The parallelization schemes we present are for distributed memory, but, when coupled with threads, they apply equally well to our machines at the box level.

In Fig. 3, we present three basic parallelization schemes. Without doubt, full replication is the best way to parallelize Monte Carlo calculations of our type. However, when the entire mesh will not fit on a single processor, the mesh must be broken up among the processors. "Full Domain Decomposition" satisfies those memory constraints and is ideal for deterministic methods, but it hinders Monte Carlo calculations too much. Specifically, communication between processors must occur during the calculation and the work in any given cell is not shared by multiple processors. The "General Domain Decomposition/Replication" scheme satisfies memory constraints *and* allows for replication of cells and reduced in-calculation communication. The key to this general scheme is to put as much of the mesh on a processor as possible and as necessary. The general scheme limits to full replication with increasing

processor capacity, and it limits to full domain decomposition with decreasing processor capacity. A disadvantage of the general domain decomposition/replication scheme is that the number of in-calculation communication channels increases with the square of the number of processors.

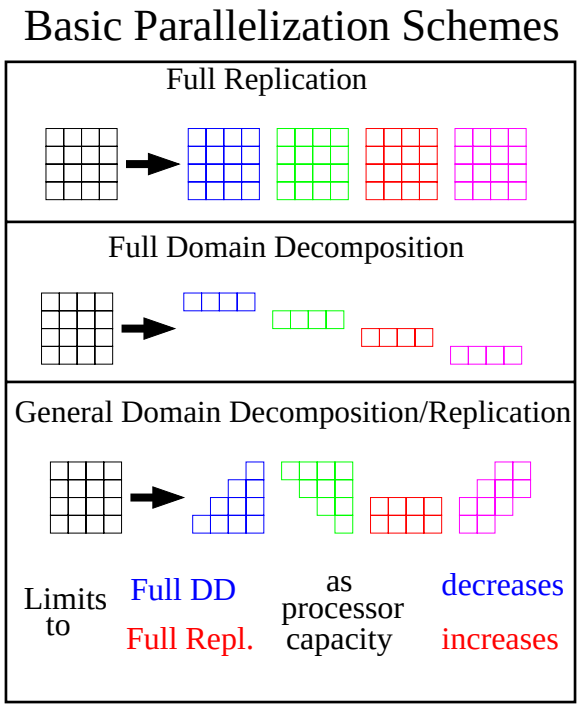

Figure 3: The three basic parallelization schemes.

Our ultimate strategy involves a hierarchical utilization of the general basic scheme [10]. The idea is to break the number of available processors into subsets and represent the entire mesh on each subset using the general basic scheme. Our two-step, hierarchical strategy retains the advantages of the basic scheme, in that important cells are replicated on every processor and in-calculation communication is reduced. Further, the poor scaling of in-calculation communication is limited to a subset, not the entire set, of processors. A schematic of our parallel strategy is in Fig. 4. Lawrence Livermore National Laboratory has a similar strategy based on the full domain decomposition basic scheme.

Our parallel strategy is not fully implemented at this time, although much of the groundwork is in place. Our parallel results in this paper are based on full replication.

New Methods

We have modified and adapted common IMC algorithms for use in MILAGRO. One glaring shortcoming of old was that the results were not reproducible when run in parallel. Successive runs on the same number of multiple processors gave statistically different solutions, both of which were different from the single-processor solution. We feel that reproducibility is essential and worth some trade-offs.

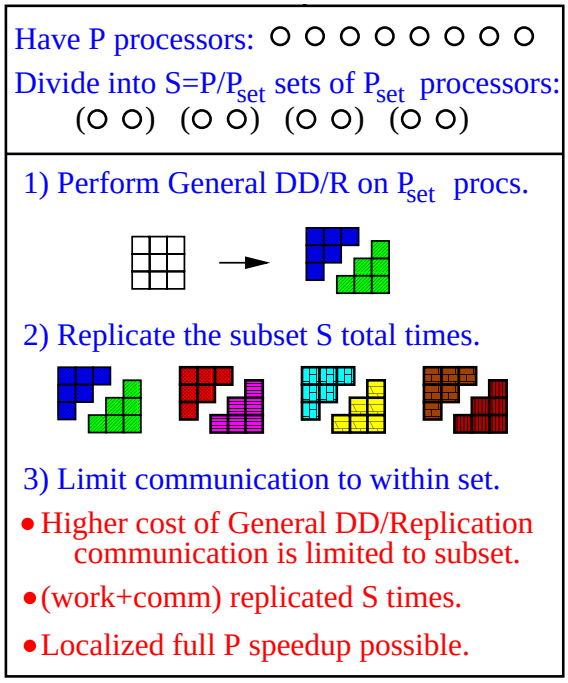

Figure 4: Our parallel strategy for IMC.

Parallel Reproducibility. An important constraint on MILAGRO is that, for the same input, it give exactly the same answer regardless of the number of processors used in the calculation. This parallel reproducibility is invaluable for testing new methods, adding new features, and debugging. Without it, each calculation performed on a different number of processors must be statistically analyzed.

Reproducibility requires that a particle have no direct relation to the processor on which it resides. A given particle should always encounter the same stream of random numbers. We meet this requirement by controlling the source serially, by assigning each particle its own random number object, and by combing each census particle in a self-dependent fashion.

MILAGRO has demonstrated parallel reproducibility in all of our test problems. We expect that at some high number of processors, roundoff errors will ruin MILAGRO's reproducibility. We theorize two possible ways of avoiding roundoff errors. The first is to simply keep the processors saturated with work, or, in other words, use no more than the optimal number of processors. The second is to accumulate crucial summations in logarithmically spaced bins, and, at the end of a timestep, sum the bins in a preordained order. The second option would be quite involved, and we would rather avoid it.

New Census Comb. The purpose of combing the census particles is to control the overall number of particles per timestep and to reduce the variance in particle energy-weights. A commonly used census comb has been developed by E. Canfield [11]. In Canfield's comb, the desired number of census particles in a particular cell is determined and the census energy divided equally among the new census particles. The attributes of the new census particles are obtained as follows: The old census particles' energy-weights are stacked end-to-end. A "comb" is produced by stacking the new census particle energy-weights. The comb is lined

up next to the stack of old particle with a random offset. Wherever a comb tooth hits the old stack, that old particle's attributes are transferred to the new census particle. The comb is shown in Figure 5. Census particles with a higher energy-weight are more likely to survive.

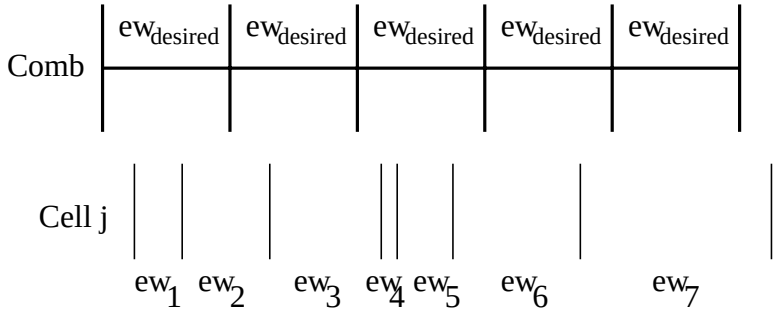

Figure 5: Canfield's census comb in a particular cell j.

Canfield's census comb has the wonderful quality of exactly conserving energy. The downside is that it is not reproducible in parallel. The comb's random number stream is not unique. Depending on how it is implemented, it may depend on the processor or it may depend on the ordering of the census particle list. It could be made reproducible by ordering the list, but, in our case, ordering is nearly impossible and certainly impractical.

The comb we use is similar to Splitting and Russian Roulette. A census particle is killed or replicated according to the ratio of the desired energy-weight to the particle's energy-weight:

$$R_p = \mathrm{Int}\Big(\frac{ew_{desired}}{ew_p} + \xi_p\Big)\,, \tag{13}$$

where particle p with energy-weight ew_p is replicated R_p times according to the desired energy-weight, $ew_{desired}$, and a random number, ξ_p, from the random number stream of particle p. This comb is fully reproducible in parallel. Unfortunately, it only statistically conserves energy. Overall variance in energy-weights is reduced, but lower energy regions that are inadequately sampled will be noisier. In a non-linear calculation, once energy is lost–even statistically–in a linearized time step, that energy is lost forever. Our test problems have shown no propagation of statistical error, only noisier low-energy regions. However, we would recommend tending toward a larger number of particles with our comb.

Simplified Tilt. In the Fleck and Cummings IMC algorithm, the time-explicit portion of the volume emission source can be quite sensitive to cell size. Therefore, the in-cell starting location of these particles is sampled according to an assumed functional dependence within the cell. Thus, more particles are sampled near the hotter edges of the cell.

The tilt we use is designed for our orthogonal cells and is much simpler than a tilt for nonorthogonal RZ geometry. Each dimension is treated independently in a fashion similar to Fleck and Canfield's tilt in one-dimensional slab geometry [12]. The functional dependence is a line running through the cell-centered T^4 with a slope based on opposite face values of T^4. Face values of T^4 are averages of neighboring cell-center values. In summary, our tilt is cell-centered, linear-discontinuous, first-order, and piecewise in dimension. Future work

includes investigating the effect of neglecting the radiation constant in the slope, i.e., using T^4 instead of aT^4.

Verification and Test Problems

Verification is the ascertainment that our code is solving the equations we think it is solving. We verify our code in two ways. First, we verify at the programming level by using Design-by-Contract™. Second, we verify in a more integral sense by comparing the code's results to analytic solutions of test problems.

The first test problems we run with new software is usually for debugging just as much as it is for verification. These are simple, degenerate test problems that are necessary but not sufficient. To test the particle tracking, we run purely streaming problems where the particles do not interact with the material. To test the interaction between radiation and material, we run a steady-state, infinite medium problem. We construct a three-dimensional XYZ box with reflecting boundaries. The initial temperatures of the material and radiation are the same, and they should stay the same. The surface source can be tested by replacing one of the reflecting walls with a Planckian surface source at the same equilibrium temperature. MILAGRO has successfully run these necessary, but not sufficient test problems.

More interesting time-dependent test problems are the Marshak Wave, with either a constant incident flux or a delta function source, and the Su and Olson non-equilibrium benchmark. We discuss the problem setups and results next.

Marshak Wave with Incident Flux. This Marshak Wave test problem considers a constant, isotropic intensity incident on a slab. The material was purely absorbing with an opacity of $100/T^3$ cm^2/g, a specific heat of 0.1 Jerks/g/keV, and a density of 3.0 g/cm^3. The incident intensity was a Planckian at 1 keV. We modeled the slab as a row of three-dimensional blocks. The y- and z-directions had one cell of thickness 0.01 cm. The x-direction had 200 cells, each of thickness 0.005 cm. The initial temperature for the system was 1e-6 keV. We ran 10,000 particles per timestep, which was a constant 0.001 shake.

The MILAGRO results at 7.4 shakes are shown in Fig. 6. The analytic results were obtained from a fourth order Runge-Kutta code written by Donald Shirk, LANL. The problem was run on the SGI/Cray ASCI Blue machine using 10 processors and full replication. The runtime is unavailable. There was, however, limited parallel speedup. The reason for the poor parallel performance is that we construct the mesh every cycle, a generality foreshadowing the interfacing to an AMR code. Plus, the number of particles and the timestep were both relatively small.

Marshak Wave with Delta Function Source. The second Marshak Wave variant is again in slab geometry, but with a delta function source of 0.01 Jerks at $x = 0$ cm and $t = 0$ shakes. Although testing the ability to model a delta function may be important, we did not feel it was relevant to verifying our code. So, we ran the problem from $t = 0.1$ shake using the analytic solution as our initial condition. The analytical data used to compare output and to produce the source was produced by Mark Gray's Analytical Test Suite[2]. The slab material had an opacity of $1/T^3$ cm^2/g, a specific heat of 0.1 Jerks/g/keV, and a density

[2]The Analytical Test Suite is an XTM application used to verify radiation transport packages.

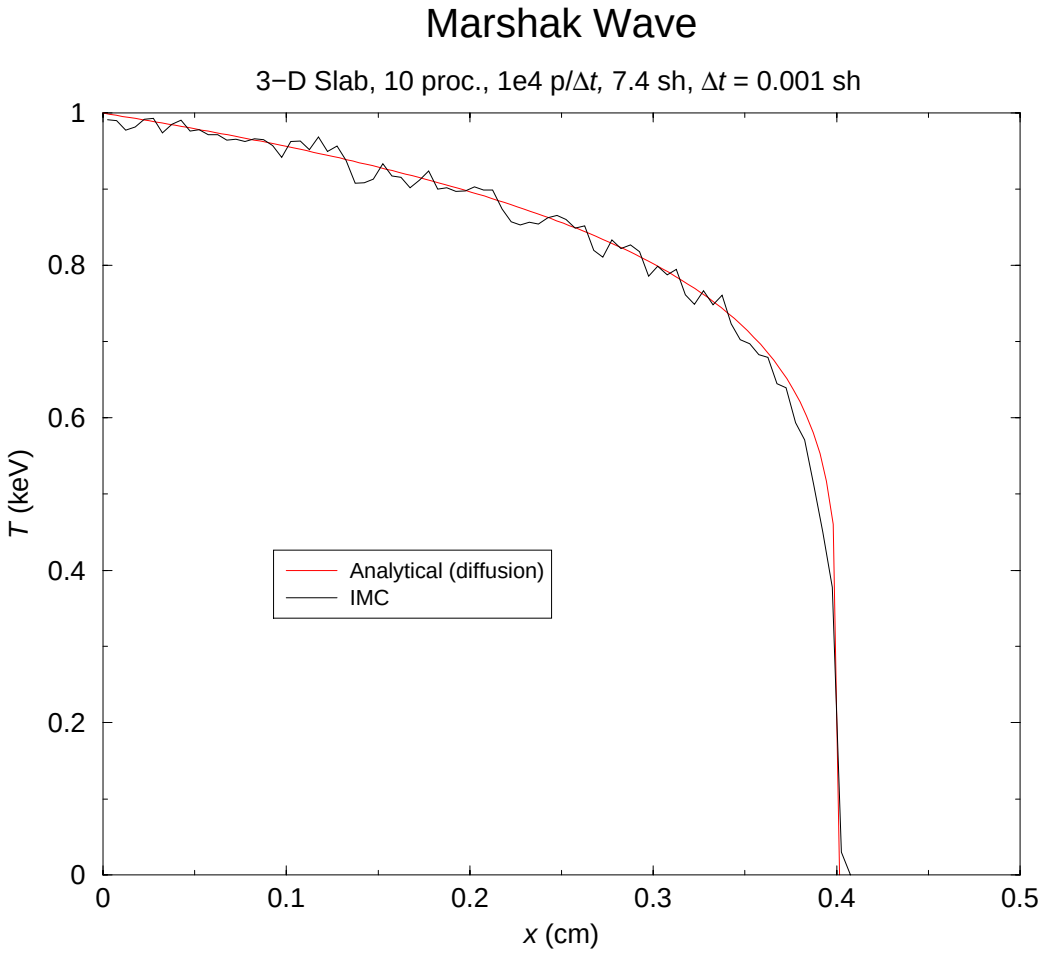

Figure 6: Marshak Wave with a constant, isotropic incident intensity.

of 3.0 g/cm^3.

For this test problem only, we have results for both MILAGRO and MILSTONE. We modeled the slab with two cells in each of the transverse directions and, in the x-direction, 200 for MILAGRO and 40 for MILSTONE. The cell size in the x-direction was 0.0025 cm. The cell size was supposed to be 0.0025 in the transverse directions, as well, but we accidentally made it ten times larger. This mistake actually made the code run faster, since there were fewer reflections. All boundaries were reflecting. We used 10,000 particles per timestep, which was a constant 0.01 shake. The initial temperatures in non-active cells were 1×10^{-5} keV. We present the results in Fig. 7, where each IMC data point is the average of the four cell values at that x location. MILAGRO and MILSTONE *should* give exactly the same answer, except that the cells are ordered differently, and the number of cells along the x-direction is different. The run-times were approximately 4.1 hours on an SGI Octane workstation.

Su and Olson Non-Equilibrium Benchmark. The final test problem is that of Su and Olson [13]. It is a cold, infinite, homogeneous slab with a finite radiation source that exists for a finite amount of time. Of the purely absorbing case and the 50% scattering case, we

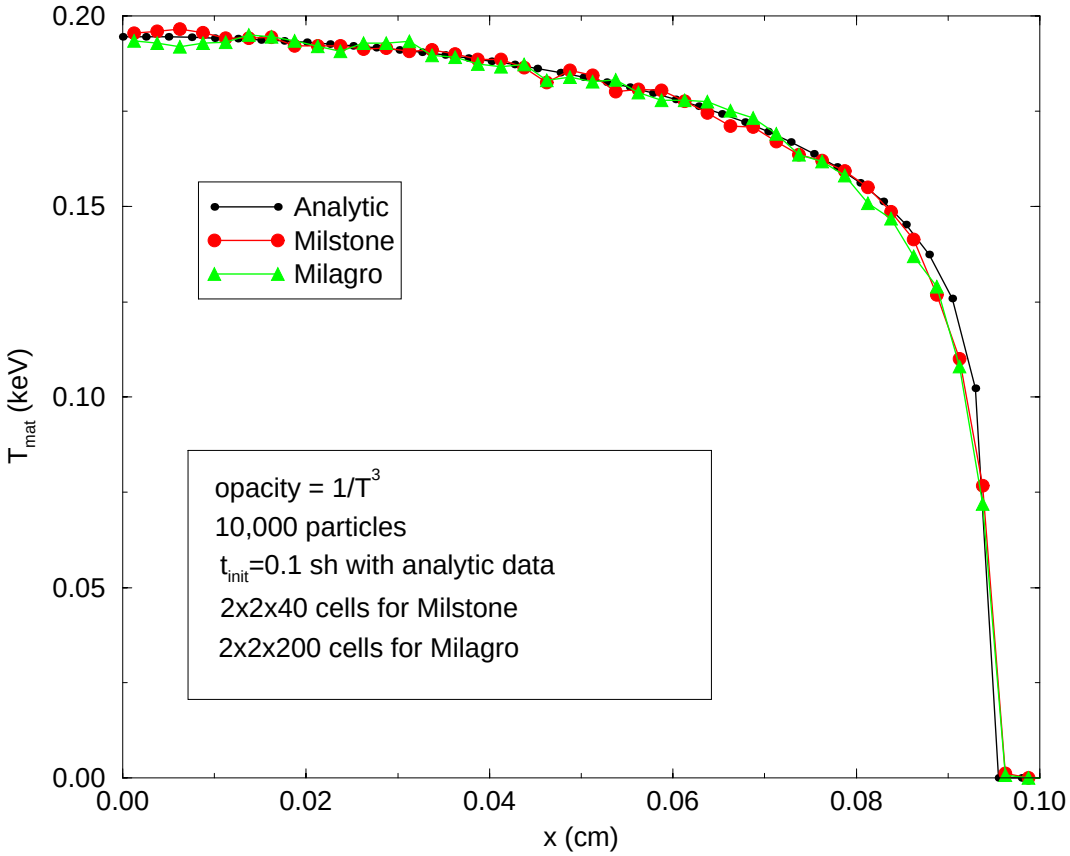

Figure 7: Marshak Wave with delta function source of 0.01 Jerks at time = 0 and $x = 0$.

only present results for the latter so as to demonstrate isotropic scattering in MILAGRO. Regardless, MILAGRO showed excellent agreement in both.

The analytic problem is scaled to a reference temperature, so some trial and error was required to find appropriate physical regimes. Somewhat surprisingly, for unit density and unit opacity, suitable initial temperatures were around 1 keV. Equalizing the initial radiation and material energies resulted in an initial radiation temperature of 1.41 keV and an initial material temperature of 1.00 keV. The specific heat varies with T_{mat}^3 and, for our particular initial conditions, was initially 0.05488 Jerks/g/keV. A suitable timestep was $0.0000\bar{3}$. We used 200 cells in the x-direction, each of thickness 0.1 cm, with reflecting boundary conditions at $x = 0$. The isotropic radiation source, normalized to unity, is located in the region $0 < x < 0.5$ cm and runs from time 0 to $0.0\bar{3}$ shakes. Since the energy in the problem increases while the source is on, we begin the problem with 5000 particles per timestep and linearly ramp up to 50,000 particles at $t = 0.\bar{3}$, or, equivalently, $ct = 100$, where $c = 300$ cm/sh. In order to compare to published results, our results had to be scaled. We multiplied the material energy by 375 and the radiation energy by four times that. The radiation and material energy plots are shown in Figs. 8 and 9, respectively. Note the spikes in the low material energy regions; they are probably noise due to our statistical comb. Note

also that the analytic solution has too few data points in some regions.

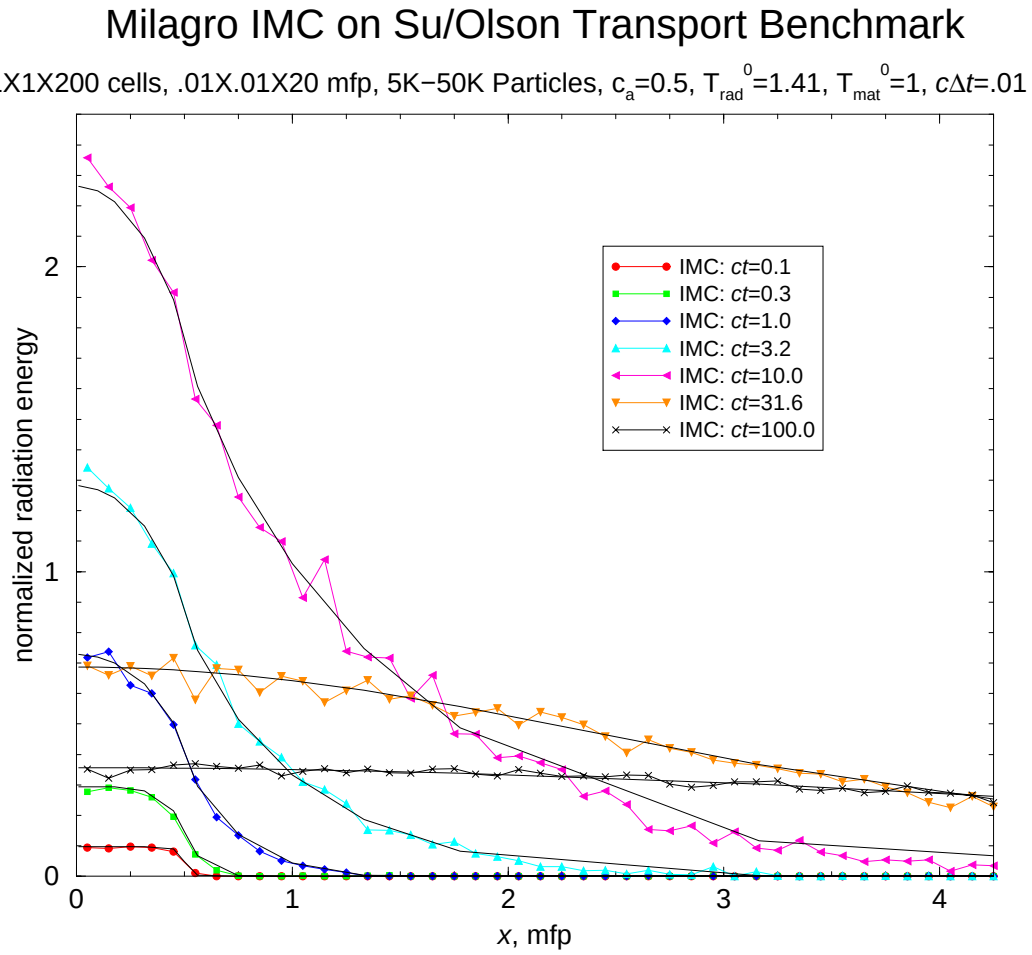

Figure 8: Normalized radiation energy for Su/Olson non-equilibrium transport benchmark.

Verification efforts will continue, but those to date have successfully shown MILAGRO to be verified.

Conclusion

We have developed a parallel, extensible, IMC code. It has full 3-D capabilities, is reproducible in parallel, has demonstrated extensibility and maintainability, and has been tested on a suite of radiation test problems. We have documented plans [10] for implementing additional capabilities that should further test the validity of our object-oriented/generic design.

Future enhancements to MILAGRO include:

1. implementation of full radiation-hydrodynamics;
2. completed parallel studies;
3. new geometries and mesh types including, AMR, RZ, degenerate hex and polyhedra;

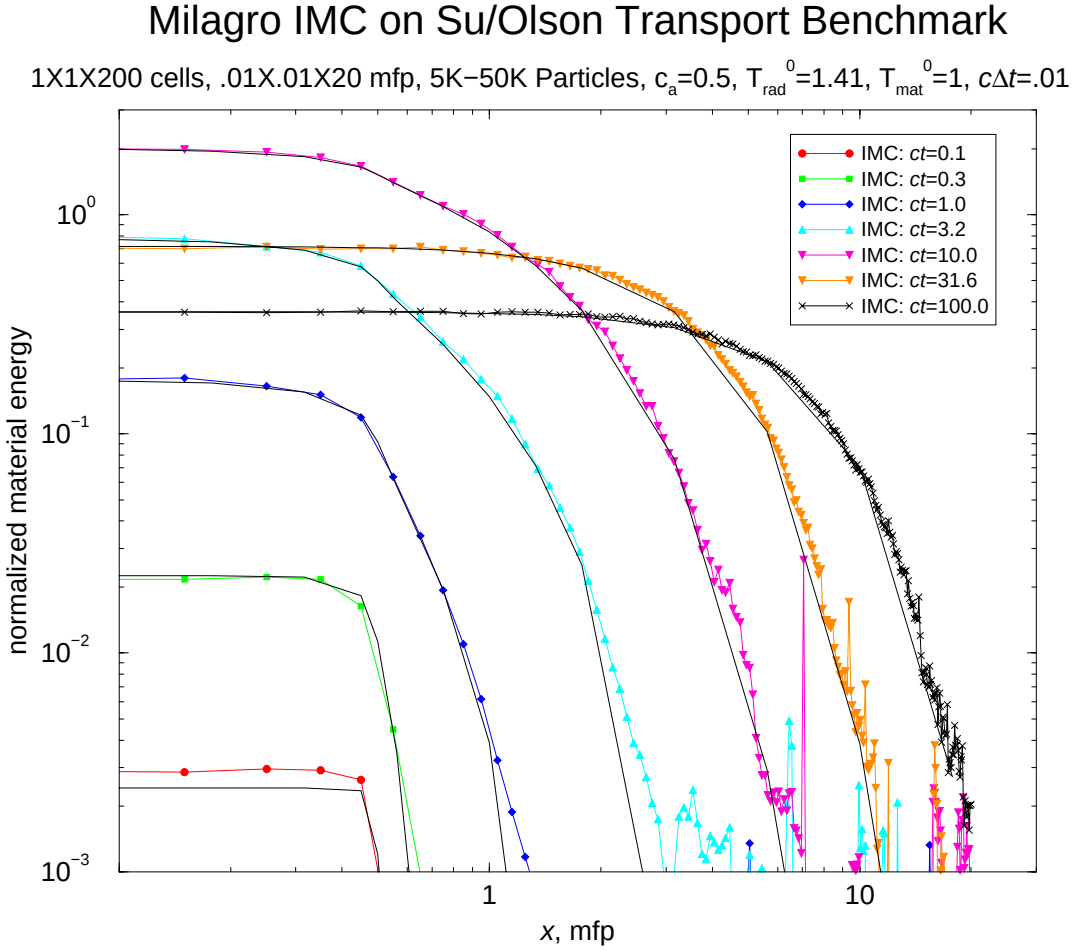

Figure 9: Normalized material energy for Su/Olson non-equilibrium transport benchmark.

4. acceleration and speedup techniques including random walk [12] and Discrete Diffusion Monte Carlo (DDMC) [14];

5. new IMC methods;

6. multi-group.

Given this list, MILAGRO can be seen as the starting point in a rigorous and extended IMC effort.

References

[1] B. Su and G. L. Olson, "A benchmark for non-equilibrium radiative transfer in an isotropically scattering medium," Technical Report LA-UR-96-1799, Los Alamos National Laboratory, 1996.

[2] B. Su and G. L. Olson, "Benchmark results for the non-equilibrium marshak diffusion problem," Technical Report LA-UR-96-488, Los Alamos National Laboratory, 1996.

[3] Kuck and Associates, *KAI C++ Compiler*. Champaign, IL, 3.2f ed., Oct. 1997. c++support@kai.com.

[4] ISO/IEC, International Standard, "Programming languages-C++," Tech. Rep. 14882, American National Standard Institute, New York, Sept. 1998.

[5] G. C. Pomraning, *The Equations of Radiation Hydrodynamics*. Oxford: Pergamon Press, 1973.

[6] J. A. Fleck, Jr. and J. D. Cummings, "An implicit Monte Carlo scheme for calculating time and frequency dependent nonlinear radiation transport," *Journal of Computational Physics*, vol. 8, pp. 313–342, 1971.

[7] E. W. Larsen and B. Mercier, "Analysis of a Monte Carlo method for nonlinear radiative transfer," *Journal of Computational Physics*, vol. 71, pp. 50–64, 1987.

[8] D. Ceperley, M. Mascagni, and A. Srinivasan, "SPRNG: Scalable Parallel Random Number Generators." NCSA, University of Illinois, Urbana-Champaign, Nov. 1997. www.ncsa.uiuc.edu/Apps/SPRNG.

[9] B. Meyer, *Object-Oriented Sofware Construction*. Upper Saddle River, NJ: Prentice Hall, second ed., 1997.

[10] T. J. Urbatsch and T. M. Evans, "Strategy for parallel Implicit Monte Carlo," Research Note XTM-RN(U)-98-018, Los Alamos National Laboratory, May 1998. LA-UR–98–2263.

[11] E. Canfield, "private communication." Aug. 1998.

[12] J. A. Fleck, Jr. and E. H. Canfield, "A random walk procedure for improving the computational efficiency of the Implicit Monte Carlo method for nonlinear radiation transport," *Journal of Computational Physics*, vol. 54, pp. 508–523, 1984.

[13] B. Su and G. L. Olson, "An analytical benchmark for non-equilibrium radiative transfer in an isotropically scattering medium," *Annals of Nuclear Energy*, vol. 24, no. 13, pp. 1035–1055, 1997.

[14] T. Urbatsch and J. Morel, "Discrete Diffusion Monte Carlo," Research Note XTM-RN(U)-97–050, Los Alamos National Laboratory, Nov. 1997.